VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

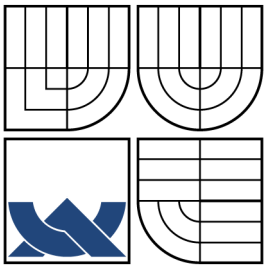
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

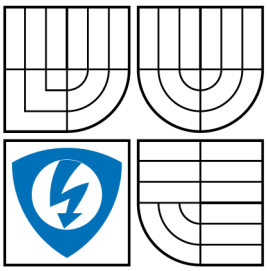
VÍCEÚČELOVÝ INFORMAČNÍ SYSTÉM ZALOŽENÝ NA PHP FRAMEWORKU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. MICHAL KAŠPAREC
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VÍCEÚČELOVÝ INFORMAČNÍ SYSTÉM ZALOŽENÝ NA PHP FRAMEWORKU

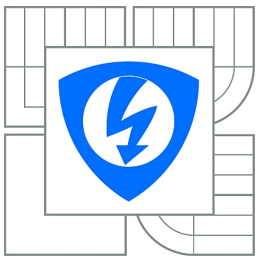
MULTIPURPOSE INFORMATION SYSTEM BASED ON PHP FRAMEWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MICHAL KAŠPAREC
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN KOUTNÝ, Ph.D.
SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Michal Kašparec ***ID:*** 115198
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Víceúčelový informační systém založený na PHP frameworku

POKYNY PRO VYPRACOVÁNÍ:

Předmětem práce bude návrh a realizace víceúčelového informačního systému. Systém bude realizovaný v některém z dostupných PHP frameworků s tím, že se uvažuje Nette nebo Zend. Jádro systému bude tvořit rozšířený RS s podporou ACL a na něj pak budou navazovat další moduly a pluginy, které bude možné jednoduchým způsobem instalovat či odinstalovávat.
Předpokládá se realizace dvou až tří modulů, které budou upřesněny v rámci konzultací.

DOPORUČENÁ LITERATURA:

[1] CASTAGNETTO, Jesus. PHP : programujeme profesionálně. 2. opravené a aktualizované vydání. Praha : Computer Press, 2002. 626 s. ISBN 80-7226-310-2.
[2] MCARTHUR, Kevin. Pro PHP: Patterns, Frameworks, Testing and More. : Apress, 2008. 349 s. ISBN 978-1590598191.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Ing. Martin Koutný, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce porovnává několik známých PHP frameworků a jejich hlavní vlastnosti. Tvorbu redakčního systému s využitím propracovaného administračního rozhraní a přístupových práv. Dále realizaci modulu pro objednání a správu chytré karty pro dopravní podnik. S využitím moderních platebních metod pomocí 3D secure. Možností dynamicky upravovat ceníky, zóny a druhy karet. Je provedena i kontrola sousedních zón aby nebylo možné vybrat nesmyslené kombinace zón.

KLÍČOVÁ SLOVA

PHP framework, Nette, MySQL, Zend, MVC, MVP, AJAX, Bcrypt, redakční systém, IDS JMK, 3D secure

ABSTRACT

This project compare several well-known PHP frameworks and their main characteristics. Creating the content management system with using sophisticated administration interface and access rights. Realization module for ordering and managing smart card for public transport company. Using modern payment methods using 3D secure. Dynamically change the price lists, zones and types of cards. It is created checking neighbor zones that can not be chosen meaningless combinations zones.

KEYWORDS

PHP framework, Nette, MySQL, Zend, MVC, MVP, AJAX, Bcrypt, CMS, IDS JMK, 3D secure

KAŠPAREC, Michal *Víceúčelový informační systém založený na PHP frameworku*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 56 s. Vedoucí práce byl Ing. Martin Koutný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Víceúčelový informační systém založený na PHP frameworku“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce Ing. Martinu Koutnému, Ph.D. za cenné rady, odborné vedení, konzultace, metodickou pomoc a podnětné návrhy při zpracování této práce.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dnešní době je snaha co nejvíce služeb řešit elektronicky. K této snaze přispívá možnost platby mnoha služeb přes online platební brány nebo přímo platební kartou. Ani dopravní podniky nechtějí být pozadu, a proto zavádějí možnosti nakupovat jízdní doklady pomocí SMS zpráv. Pomalu se začíná objevovat i metoda, kdy je možné nahradit obyčejné papírové jízdenky jejich modernější variantou. Řeč je o moderních čipových kartách a možnost je online, z pohodlí domova, nabíjet a zakupovat na ně nové kupóny, případně prodlužovat jejich platnost. K tomuto účelu je ale potřebné, aby bylo vyhotoveno nějaké online prostředí, kde se dají takové akce provádět. To vedlo k otázce, jak takový systém řešit. Nejdříve je potřeba určit v čem bude vhodné začít programovat a poté zjistit, co všechno musí takový systém obstarávat.

V tom důsledku bylo popsáno a porovnáno několik PHP frameworků. Výběr frameworku není vždy úplně snadný, a proto bylo potřeba porovnat některé známé a používané PHP frameworky. Uvažovat se měly PHP frameworky Nette a Zend, které byly důkladně porovnány. Poté bylo realizováno jádro pro redakční systém, včetně databáze a správy uživatelů. Jako hlavní modul pro realizaci byl navrhnut modul pro dopravní podnik.

1 CHARAKTERISTIKA FRAMEWORKU

Framework je balík naprogramovaných tříd a objektů, který definuje, jak se mají používat, upravovat nebo vytvářet nové třídy, objekty a modely.

1.1 K čemu slouží?

Framework má za úkol převzít typické problémy dané oblasti, a tím usnadnit vývoj tak, aby návrháři a vývojáři mohli pracovat pouze na svém zadání. Při používání frameworku je potřeba dodržovat určitá pravidla, kvůli základní funkčnosti frameworku a některá doporučená pravidla, aby se dalo orientovat v cizích zdrojových kódech.

Objevují se námitky, že při použití frameworku bude kód pomalý, či jinak neefektivní, a že čas, který se ušetří jeho použitím, se musí věnovat nastudování frameworku. Nicméně při jeho opakovaném nasazení nebo ve velkém projektu dojde k výrazné úspoře času. O proti čistému PHP je v menších projektech nevýhodou větší paměťová náročnost a s tím i spojená doba na zpracování požadavku.

1.2 Dostupnost, licence

Existuje několik typů frameworků:

- „Soukromé“ frameworky – vytváří je pouze jeden programátor a pouze pro vlastní účely,
- „firemní“ frameworky – vytváří je tým více programátorů v jedné firmě,
- „open-source“ frameworky – vyvíjí je nezávisle na sobě několik vývojářů a vyvíjí dohromady jeden framework.

Velká nevýhoda soukromých frameworků je, že nad tvorbou frameworku musí programátor strávit spoustu času než ho vytvoří, otestuje a opraví chyby. Firemní frameworky většinou zůstávají v dané firmě a dále se nešíří. Open-source frameworky jsou nejrozšířenější a vzniká kolem nich komunita vývojářů, kteří využívají tento framework pro své účely.

Open-source frameworky jsou vyvíjeny pod licencí GNU-GPL [3]. Tato licence umožňuje, že software, který je pod touto licencí vydán, může být volně distribuován, spouštěn, měněn, kopírován, studován a zlepšován. Je možné ho jakkoli přizpůsobit pro své účely, přidávat své nápady a poté tato vylepšení zveřejnit ostatním. Nemusí se za něj nic platit a není nutné nikoho žádat o povolení.

2 TYPY FRAMEWORKŮ

Na základě studia nejpoužívanějších frameworků [12] bylo vybráno několik zajímavých frameworků [8], které byly důkladně prostudovány a srovnány v tab. 2.1.

Tab. 2.1: Některé známé frameworky

Framework	PHP5	Multi BD	MVC	AJAX	Moduly
Kohana	ANO	ANO	ANO	ANO	ANO
CakePHP	ANO	ANO	ANO	ANO	ANO
CodeIgniter	ANO	ANO	ANO	NE	NE
Prado	ANO	ANO	ANO	ANO	ANO
Zend	ANO	ANO	ANO	ANO	ANO
Nette	ANO	ANO	ANO	ANO	ANO

2.1 Srovnání nejznámějších frameworků

2.1.1 Kohana

Tento MVC framework je nyní dostupný ve verzi 3.3.0. Kohana framework je určen pro PHP 5 [11]. Jedná se o přepracovaný framework CodeIgniter, od kterého se liší hlavně vyšší verzí PHP, a že jej vyvíjí pouze komunita.

- Využívá ORM knihovnu. Podporuje databáze MySQL [9], PostgreSQL, MsSQL a PDOSqlite.
- Obsahuje několik pluginů, jako je Gmaps pro integraci Google Maps, archive pro vytváření archivů, payment pro internetové platby a další.
- Zahrnuje 13 knihoven a 20 pomocných funkcí (helpers).

2.1.2 CakePHP

Framework je dostupný ve verzi 2.2.3. CakePHP je určen pro PHP 4 i pro PHP 5. Patří mezi nejvíce rozšířené frameworky, a proto je na internetu k dispozici i mnoho českých návodů.

- Používá ORM knihovnu. Možnost připojení na databáze MySQL, PostgreSQL, ADOdb, Firebird DB2, MSSQL, Oracle, SQLite, ODBC.
- Obsahuje komponenty pro práci s emaily, autentifikaci a mnoho dalších.
- Velmi dobře zpracovaná validace, která se provádí v modelu.
- Kromě standardních helperů zde můžeme nalézt také helpery pro AJAX či RSS.

2.1.3 CodeIgniter

Framework je aktuálně dostupný ve verzi 2.1.3. Je určen pro PHP 4 i pro PHP 5. Jedná se o velmi jednoduchý a intuitivní framework, který je vyvíjen firmou a ne komunitou. Má velmi dobrou dokumentaci.

- Podpora databází MySQL, MSSQL, PostgreSQL, Oracle, SQLite, ODBC. Pro práci s databází používá upravenou třídu Active Record. Tato „ORM“ třída však neumí pracovat s relacemi.
- Najdeme zde několik užitečných pluginů pro práci s Excelem, CAPTCHA, generátory PDFS či geografické lokátory, které podle IP adresy určují pozici.
- Validace probíhá v kontroléru, kde je možné definovat jednoduchým zápisem i složitější pravidla.
- Modely, které chceme v dané metodě používat, je třeba načíst pomocí funkce load.

2.1.4 Prado

Aktuálně se nachází ve verzi 3.2.0. Je určen pouze pro PHP 5. Prado není klasický PHP framework. Je založený na komponentovém přístupu a programování událostí. Jeho vzorem při vývoji nebyl Ruby on Rails jako u většiny PHP frameworků, ale inspiroval se spíše ASP.NET.

- Podporuje databáze MySQL, PostgreSQL, SQLite, MSSQL, Oracle.
- Pro přístup do databáze je možné volit mezi PDO, Active Record nebo kompletním objektově mapovým schématem SQLMap.
- Obsahuje validaci základních datových typů, dále je také možnost přidat k formuláři kontrolu CAPTCHA.

2.1.5 Zend

Patří mezi nejznámější PHP frameworky. Aktuální verze je 2.0.5. Je vyvíjen přímo firmou Zend, která vyvíjí jádro PHP. Celý framework je objektově orientovaný a implementovaný v PHP 5. Zend framework jako celek využívá návrhový vzor MVC.

- Podporuje databáze MySQL, MSSQL, SQLite, Oracle, PostgreSQL, IBM DB2.
- Pro přístup do databáze využívá PDO.
- Součástí Zendu je celá řada modulů, např. ověřování práv, posílání emailů nebo tvorba PDF dokumentů.
- Lze využít šablonovací systém SMARTY[1].

[1]SMARTY - je komplexní a funkčně velmi dobře vybavený šablonovací systém pro PHP.

2.1.6 Nette

Jedná se o český PHP framework, jehož autorem je David Grudl. Aktuální stabilní verze je 2.0.7 a je psaný v PHP 5 s plným využitím objektově orientovaného programování. Původně byl psán jako „soukromý“ framework, avšak později byl uvolněn jako open-source. Díky tomu má možnost se rychleji a lépe rozšiřovat o nové funkce.

- Pro práci s databází se může použít integorvaná funkce Database, a nebo se může zvolit jiná knihovna (např. NotORM, Doctrine2 aj.).
- Používá vlastní šablonovací systém Latte.
- Konfigurace probíhá úpravou souboru .neon.

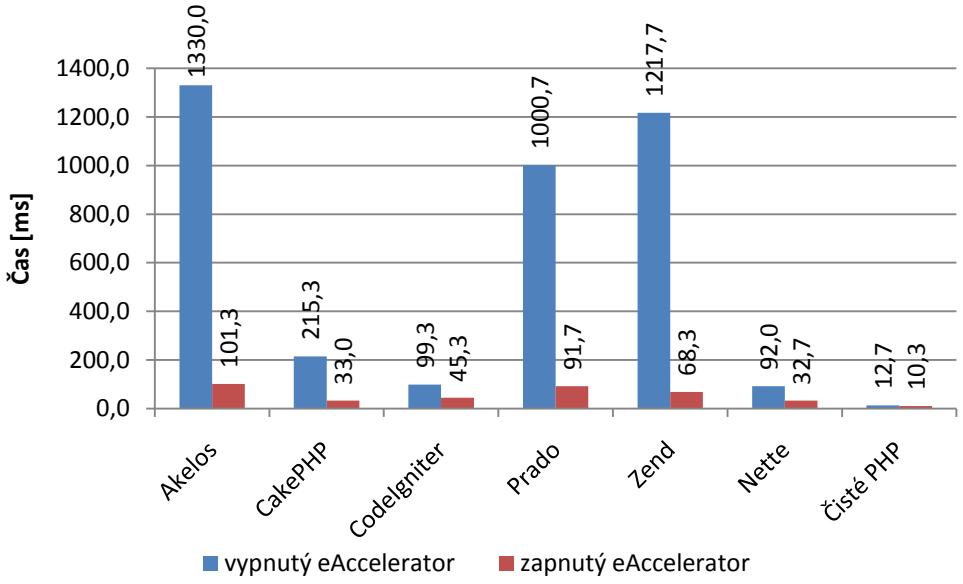
Hlavní výhoda PHP frameworků, které nepodporují PHP 4, ale pouze PHP 5 je, že obsahují zcela nový objektový model, který přináší možnost plnohodnotného objektově orientovaného návrhu včetně abstraktních rozhraní.

2.1.7 Srovnání rychlostí

Hodnoty, uvedené v následujícím grafu, jsou průměry hodnot získané z testování [2]. Test byl složen ze čtyř kroků (30x zobrazení členů, 30x zobrazení uživatelů, 30x editační formulář, 30x update dotaz na databázi MySQL). Do testu bylo pro srovnání zahrnuto i čisté PHP.

Z grafu na obr. 2.1 je vidět, že problém s rychlostí má několik PHP frameworků, jejichž průměrný čas překračoval jednu sekundu. Překvapivě mezi ně patří i velice oblíbený Zend framework. Při zapnutí eAcceleratoru[2] se časy všech frameworků značně zlepšily a už by nebyl problém s jejich provozem.

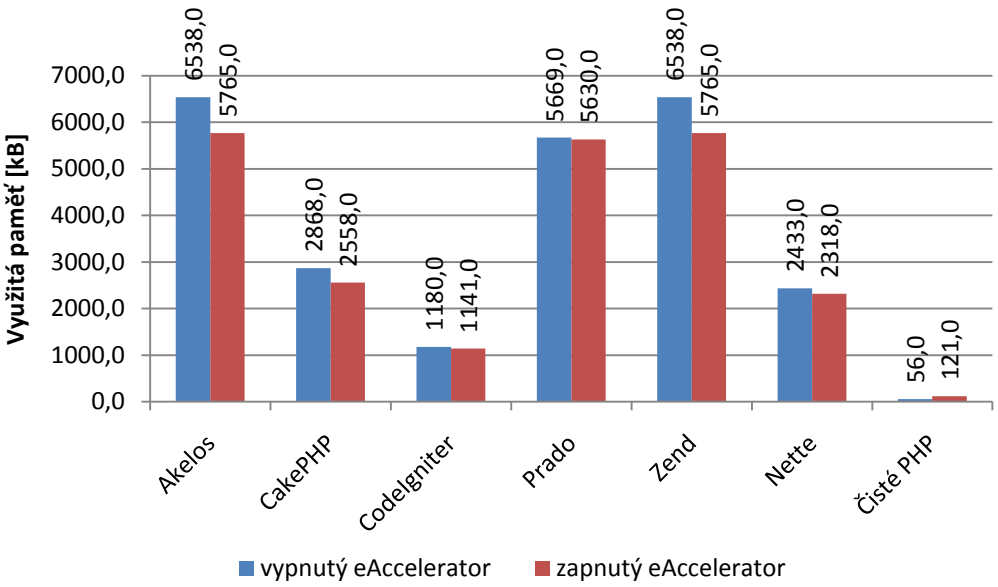
[2]eAccelerator je velmi užitečný nástroj pro „zrychlení“ PHP. Funguje na principu vytváření mezipaměti (cache) PHP skriptů.



Obr. 2.1: Porovnání rychlostí jednotlivých frameworků.

2.1.8 Srovnání paměťových nároků

Kromě rychlosti frameworků hraje významnou roli také jejich paměťová náročnost. Jak je vidět na obr. 2.2, tak zapnutí eAcceleratoru nemá žádný význam. Většina frameworků se drží okolo 5MB využité paměti.



Obr. 2.2: Využití paměti u jednotlivých frameworků.

3 SROVNÁNÍ NETTE VS. ZEND

Vzhledem k tomu, že se mají uvažovat pouze php frameworky Nette a Zend, tak uděláme detailnější porovnání, hlavně výkonnostní.

3.1 Průběh testování

Testování probíhalo na dvou počítačích, aby měřící program neovlivňoval test. Na prvním notebooku běžel testovací program Apache JMeter 2.9 a na druhém notebooku byly umístěny samotné testované stránky. Konfigurace notebooku s webovým serverm:

- Procesor: Intel Core i5-2410M @ 2.3 GHz
- Paměť: 8 GB (PC10700 SO-DIMM (DDR3–1333))
- HDD: 640 GB, 7200 otáček; SSD 128 GB

Počítače byly propojeny místní sítí 1 Gbit/s. Program JMeter běžel pod operačním systémem Windows 7 Professional 64-bit. Testované stránky běžely pod operačním systémem Windows 7 Ultimate 64-bit s konfigurací:

- PHP 5.4.7 (x86),
- Apache 2.4.3 (x86),

- PHP 5.4.5 (x64),
- Apache 2.4.4 (x64).

Pro simulaci produkčního provozu byl použit ještě operační systém Linux v distribuci Debian 7.0 64-bit s konfigurací:

- PHP 5.4.4 (x64),
- Apache 2.2.22 (x64).

První test zobrazuje kolik spotřebuje framework na svůj vlastní provoz a kolik času zabere vykreslení výchozí stránky ze „sandboxu“. Čas se průměruje ze třiceti zobrazení.

Tab. 3.1: Srovnání času Nette vs. Zend

PHP framework	**Nette**	**Zend**
Win x86 software	98 ms	181 ms
Win x64 software	94 ms	170 ms
Debian x64 software	44 ms	69 ms

Tab. 3.2: Srovnání paměti Nette vs. Zend

PHP framework	**Nette**	**Zend**
Win x86 software	3.3 MB	3.3 MB
Win x64 software	5.1 MB	5.2 MB
Debian x64 software	4.4 MB	4.5 MB

Paměťové nároky jsou u obou frameworků na všech testovaných systémech velmi podobné. Na stejném serveru jsou hodnoty téměř totožné a rozdíl mezi jednotlivými typy použité platformy je nejspíše způsoben v knihovnách samotného serveru.

Časové nároky jsou o poznání rozdílnější. Nastavení Apache serveru a použitých modulů v PHP 5 byly stejné jak na OS Windows tak na OS Debian. Rozdíl mezi 64-bitovou a 32-bitovou verzí na OS Windows není prakticky žádný. Ovšem při srovnání běhu na linuxovém jádře nebo na Windows je rozdíl velký. U Nette frameworku se čas na zpracování dotazu a vykreslení stránky snížil o polovinu, kdežto u Zend frameworku je rozdíl 60 %.

Druhý test se provádí s připojením na databázi a zobrazení deseti článků.

Tab. 3.3: Srovnání času s databází Nette vs. Zend

PHP framework	**Nette**	**Zend**
Win x86 software	112 ms	204 ms
Win x64 software	115 ms	209 ms
Debian x64 software	66 ms	93 ms

Tab. 3.4: Srovnání paměti s databází Nette vs. Zend

PHP framework	**Nette**	**Zend**
Win x86 software	3.7 MB	4.0 MB
Win x64 software	5.4 MB	5.7 MB
Debian x64 software	4.8 MB	5.0 MB

3.2 Nette framework

Nette Framework je výkonný framework pro pohodlné a rychlé vytváření kvalitních a moderních webových aplikací v PHP 5. Eliminuje bezpečnostní rizika, podporuje AJAX, DRY, KISS, MVC a znovupoužitelnost kódu.[5]

3.2.1 Zabezpečení

Cross-site scripting (XSS)

Cross-site scripting je metoda narušení webových stránek zneužívající neošetřených výstupů. Útočník pak dokáže do stránky podstrčit svůj vlastní kód a tím může stránku pozměnit nebo dokonce získat citlivé údaje o návštěvnících. Proti XSS se lze bránit jen důsledným a korektním ošetřením všech řetězců.

Nette Framework přichází s revoluční technologií *Context-aware escaping*, která vás provždy zbaví rizika XSS. Všechny výstupy totiž ošetřuje automaticky. Pokud např. zadáte:

```
$message = 'Šířka 1/2"'
```

framework vygeneruje kód:

```
&quot;Šířka 1\/2\&quot;&quot;
```

3.2.2 Cross-site request forgery (CSRF)

Cross-site request forgery spočívá v tom, že přimějeme uživatele navštívit stránku, která skrytě vykoná útok na webovou aplikaci, kde je uživatel právě přihlášen. Lze tak například pozměnit nebo smazat článek, aniž by si toho uživatel všiml. Proti útoku se lze bránit generováním a ověřováním autorizačního tokenu.

Nette Framework obranu formulářů před útokem CSRF zjednodušuje tak, že snadnější to být nemůže. Stačí totiž jediný příkaz:

```
$form->addProtection();
```

Další zabezpečení

Nette obsahuje ještě spoustu dalších zabezpečení. Mezi další významné patří např. *URL attack, control codes, invalid UTF-8*. Tyto zabezpečení obstarávají důsledné ošetřování všech vstupů na úrovni jednotlivých bajtů. Tím Nette brání útočníkovi podstrčit webové aplikaci *škodlivý* vstup.

Za zmínku ještě stojí *session hijacking, session stealing a session fixation*. Nette si samo nakonfiguruje PHP na serveru tak, aby útočník nemohl zcizit nebo podstrčit

uživateli své session ID a díky tomu získat přístup do webové aplikace, aniž by znal přihlašovací údaje uživatele.

3.2.3 Ladící nástroje

Zajímavou výhodou je debugovací nástroj zvaný „laděnka“. V klasickém případě, kdy je udělána při programování chyba, se zobrazí chybová hláška, kde méně znalý programátor těžce dohledává chybu. Při zapnutí debugovacího nástroje je chování mnohem přívětivější k vyhledání dané chyby. V objektu, kde nastala chyba, zobrazí část PHP souboru s jeho kódem a zvýrazní kód, kde se chyba vyskytla. Defaultně je nastaveno, že framework sám pozná, zda se nachází na vývojovém serveru, a nebo na produkčním. Podle toho „laděnku“ zapne nebo vypne. Pokud je však nastaveno přímo, aby byla zapnuta, tak může nastat bezpečnostní riziko při nasazení projektu na „ostrý“ provoz. V takovém případě může potenciální útočník prohlížet přímo zdrojové kódy.

4 REDAKČNÍ SYSTÉM

„Redakční systém“ je aplikace, která spravuje data a informace různého charakteru a obsahu. Zároveň se stará i o jejich efektivní využití a zobrazení na některém z předpřipravených výstupů. [15]

4.1 Úvod

Redakční systém (RS) pracuje s daty na vstupu i na výstupu. Jako vstupní rozhraní slouží například nějaký textový prvek nebo může mít i nějakou multimediální podobu. Pro jednoduchost ovládání se na textové prvky přidávají rozšíření, podobající se vzhledu známých textových editorů (např. Microsoft Office). Jako výstup slouží nějaké předpřipravené šablony, které se dají z velké části upravovat pomocí stylů. Vzhledem k tomu, že prezentace dat je zajišťována automaticky dle nastavení při prvotní instalaci nebo poté správcem redakčního systému, mají redaktoři jediný úkol – plnit systém daty. S tím souvisí, že vyspělé RS umožňují vytvořit hierarchii přístupových práv, ale k tomu se dostaneme později.

Vzhledem k tomu, že většina RS obsahuje více funkcí (modulů), které je potřeba obsluhovat, je výhodné (nutné) zřídit administrační rozhraní (beckend). V administračním rozhraní se nastavuje globální konfigurace a mělo by zároveň umožňovat plnit jednotlivé části daty. Toto rozhraní je chráněno autorizací. Jakýkoliv zásah cizí entity bez důkladné autentizace může být katastrofální.

4.2 Základní funkce

Co vše by měl umět redakční systém? Tato otázka je lehká i těžká zároveň. Na tuto otázku je několik možností jak odpovědět, přičemž některé jsou snadné a některé obtížné.

Redakční systém by měl umět zobrazovat a spravovat články. Správou článků se rozumí přidávání, upravování a mazání. Tím by se dalo říct, že je to hotový RS, jen neobsahuje žádné jiné funkce, což v dnešní době jen tak někomu nebude stačit.

Výběr konkrétních modulů co by měl RS obsahovat je složitý, protože vždy záleží na konkrétních požadavcích, ale i tak lze vytknout několik funkcí bez kterých by se RS neměl vyskytovat.

4.2.1 Důležité funkce

Souhrn funkcí, které by měl kvalitní RS obsahovat:

- správa uživatelů (alespoň základní schopnost přidávat/ubírat uživateli práva),
- komplexní článkový management (přidávání, úprava a mazání článků),
- modul komentářů k článkům,
- modul novinek,
- možnost definice rubrik,
- vyhledávací modul,
- anketní modul,
- modul správy souborů.

Pokud má být RS na větší projekty, kde se o obsah bude starat více lidí, měla by se zvážit implementace pokročilejší správy uživatelů a navrhnout Access Control List (ACL). Je třeba zbytečné, aby redaktor měl přístup k nejdetajlnějšímu nastavení a zabezpečení systému nebo správě uživatelů, které by měl mít jen správce systému (administrátor). Toto ovšem slouží spíše k lepšímu komfortu a na funkci samotného RS to nemá velký vliv. Nicméně je to z hlediska bezpečnosti nejlepší možná volba.

4.2.2 Vhodné funkce

Další funkce, které se můžou hodit, ale nejsou nezbytné:

- reklamní modul,
- statistika přístupů a čtení článků,
- fotogalerie,
- diskuzní fórum nebo návštěvní kniha,
- volby typu vkládání dat (např. Texy!, WYSIWYG, HTML).

Velkou výhodou je možnost využití přepínání mezi metodami vkládání dat. Mezi laiky je velmi oblíbená metoda pomocí WYSIWYG editoru. Avšak rozšíření Texy! [4] má spoustu výhod a je vhodné i pro zkušené uživatele, kteří preferují zápis pomocí HTML.

4.3 Výhody a nevýhody

Hlavní výhodou RS je jejich značná automatičnost a efektivnost práce s daty.

Další výhoda je jednoduchost pro uživatele při vkládání dat, která může být programátorem dovedena k luxusu pomocí speciálních editačních nástrojů. Pomocí WYSIWYG editoru je umožněno pracovat s články jako ve standardním textovém editoru. Jako nevýhodu těchto editorů lze uvést značné znepřehlednění kódu a někdy

i ztráta HTML validity systému. Jako výhodu lze uvést možný výběr mezi více metodami psaní v případě, že ji RS obsahuje.

V neposlední řadě lze za velké plus považovat i velký rozsah funkcí.

Všechno má i své nevýhody. U RS je to spíše nepřímá nevýhoda vyplývající z konkrétní situace. Jako příklad rozšiřování systému vlastními silami bude velice pracné.

4.4 Cena

Existují projekty, které nabízí RS zcela zdarma a jsou šířeny pod licencí GNU-GPL. Jako takové se dají uvést známé RS:

- Joomla!
- Drupal
- WordPress

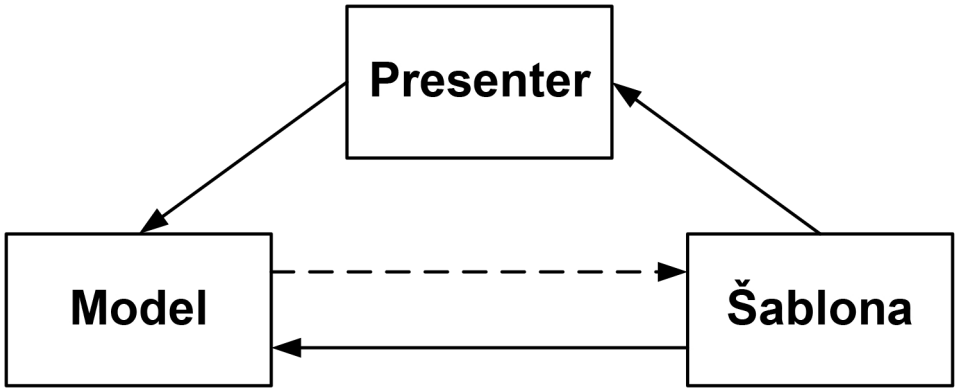
Těchto redakčních systémů je opravdu velké množství a liší se jak kvalitou, tak i množstvím funkcí. Více příkladů naleznete ve zdrojích [10].

Naopak existují firmy, jenž si nechávají za svoje RS platit od malých částek až po obrovské finanční sumy. Výběr je opravdu pestrý a dá se říci, že si vybere každý. Je ovšem důležité vybírat důkladně, protože v tomto odvětví neplatí pravidlo „drahý = kvalitní“. I spousta open-source projektů je velice kvalitních a proto sklízí velký úspěch. V dnešní době některé volně dostupné RS s vhodně použitými pluginy dosahují takových kvalit, že leckdy překonávají některé placené. Placené systémy se vyplatí, pokud potřebujeme opravdu profesionální, na míru šitý RS, který přesně odpovídá požadavkům dané problematiky.

Nastává ještě možnost, že potřebujete velmi specifický RS, kde mezi open-source produkty není vhodný kandidát nebo je svými vlastnostmi blízko, ale pořád mu nějaké důležité chybí. Upravovat open-source systémy sice můžete, ale není to nikdy snadné. Kvůli jakémukoli zásahu musíte pochopit logiku a strukturu velké části systému. Nechat si vytvořit redakční systém je celkem nákladné, proto je v takovém případě vhodné uvažovat o vlastní tvorbě, kde je možné všechno naprogramovat podle svých představ.

5 VLASTNÍ REALIZACE

Realizace jádra probíhala s ohledem na bezpečnost celé aplikace. Samozřejmostí byl vývoj podle vzoru Model-View-Presenter (MVP) [1] [6]. Činnosti modelů jsou omezeny pouze na zpracování dat z databáze a přípravu na ukládání dat do databáze, a jak vyplývá z MVP vzoru, model neví o tom, že nějaké view (šablony) a presentery existují.



Obr. 5.1: Diagram MVP

Šablony byly navrženy tak, aby získávaly data z modelu přes presenter, kde se předtím zpracují. Logika v šablonách byla omezena pouze na iterace, if, else.

Presenter spojuje šablonu s modelem a realizuje různé uživatelské akce, kontroluje oprávnění uživatelů, zpracovává formuláře a jejich validitu a spoustu dalších. Přes presentery získávají šablony data z modelů, kde mohou být před předáním šabloně různě upravena a vyfiltrována určitá data. Využívají se zde často používaná komponenta na stránkování, kde se z modelu získají všechna data a ty se pomocí určitých pravidel rozdělí na stránky a šabloně se předávají pouze data z aktuální stránky.

5.1 Návrh databáze

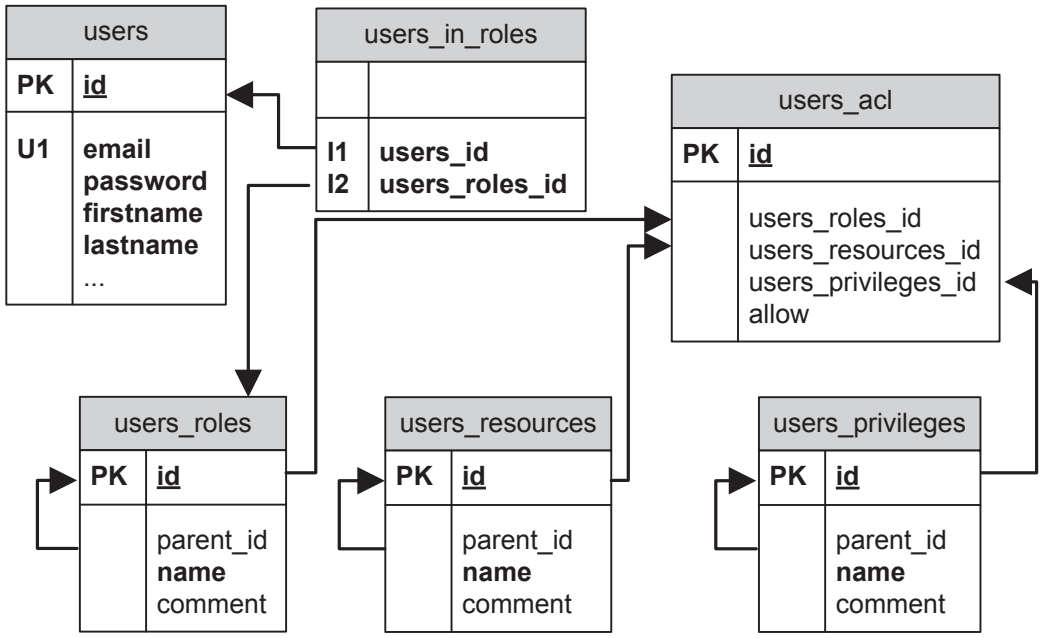
Velký důraz je kladen na zabezpečení uživatelských účtů – hlavně hesel.

5.1.1 Tabulky uživatelů

Celá databáze byla navržena s ohledem na zabezpečení celé aplikace. Jednotlivé tabulky, které se starají o uživatelské role a oprávnění, byly navrhnuty tak, aby

[1] MVP je obdoba MVC, kde *presenter* v Nette je totéž, co *controller* v jiných frameworcích.

se daly jednoduše přidávat, odebírat nebo měnit role, zdroje a oprávnění. Návrh databáze uživatelů je znázorněn na obr. 5.2.



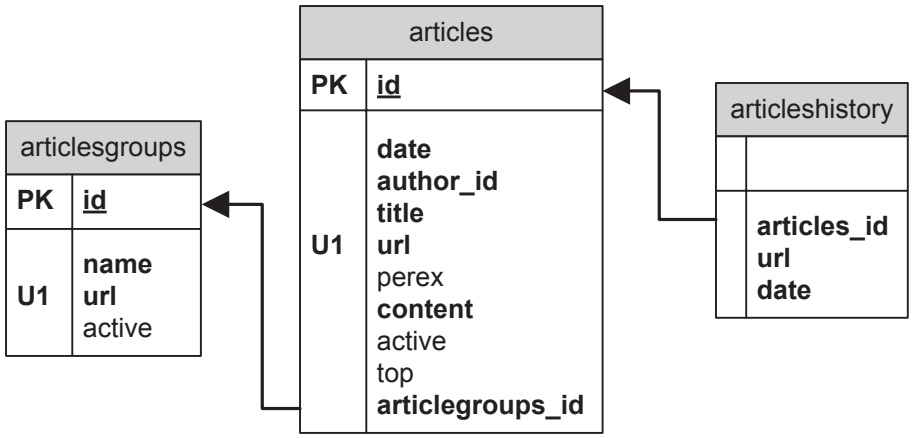
Obr. 5.2: Návrh tabulek uživatelů

Jelikož byla zvolena metoda dynamických rolí, musela být přepracována i třída ACL. V tabulce `users_roles` jsou definovány uživatelské role, které mohou být přiřazovány jednotlivým uživatelům. Každý uživatel může mít i více rolí. Jednotlivé role od sebe mohou dědit s tím, že doporučená hierarchie je postupovat od rolí s nejnižším oprávněním po ty s vyšším. Hlavní důvod tohoto postupu je samozřejmě bezpečnost. Tabulka zdrojů (`users_resources`) určuje, ke kterým částem aplikace může uživatel přistupovat (např. administrační rozhraní). Jejich formát je přesně dán samotným systémem – každý presenter má svůj záznam v databázi. Tabulka `users_privileges` definuje, jaké akce může uživatel vykonávat. Jako příklad je vhodné hlasování v nějaké anketě. Nepřihlášený uživatel (guest) má oprávnění pouze anketu vidět, ale nemá žádnou možnost nějak s ní pracovat. Zatímco přihlášený uživatel (authenticated) má zpřístupněnou akci hlasování (vote). A nejvyšší pravomoc bude mít uživatel v roli oprávněné ke všem úkonům u anket (poll_admin nebo nejvyšší admin).

V tabulce `users_acl` jsou jednotlivým rolím přidělována práva, tak aby mohl uživatel v dané roli přistupovat jen tam, kam může. Tabulka uživatelé v rolích (`users_in_roles`) přiřazuje jednotlivým uživatelům role. Každý uživatel může mít libovolný počet rolí. Výchozí role pro každého „návštěvníka“ je role *guest*. Po přihlášení se jeho role změní na *authenticated*.

5.1.2 Tabulky článků

Tabulky na správu článků (obr. 5.3) jsou vytvořeny tak, aby bylo možné články třídit do více kategorií a zaznamenávaly se změny Unique Resource Locator (URL) adresy článku do tabulky `articleshistory` z důvodu indexování stránek roboty. Funkce je jednoduchá – pokud nenajde podle URL článek v tabulce `articles`, tak začne vyhledávat v tabulce `articleshistory` a v případě shody podle ID článku článek zobrazí. Pokud nastane, že se v tabulce `articleshistory` k jedné URL najde více záznamů, tak rozhoduje datum a vezme se nejnovější článek. V tabulce článků není možné mít dva záznamy se stejnou URL. URL se vytváří z názvu článku a vždy se kontroluje, zda již není použita u jiného článku, v případě shody se na začátek řetězce přidá ID článku. Pro bezpečnost je v tabulce `articles` nad sloupcem `url` přidán unikátní index, který kontroluje unikátnost URL na úrovni databáze.



Obr. 5.3: Návrh tabulky článků

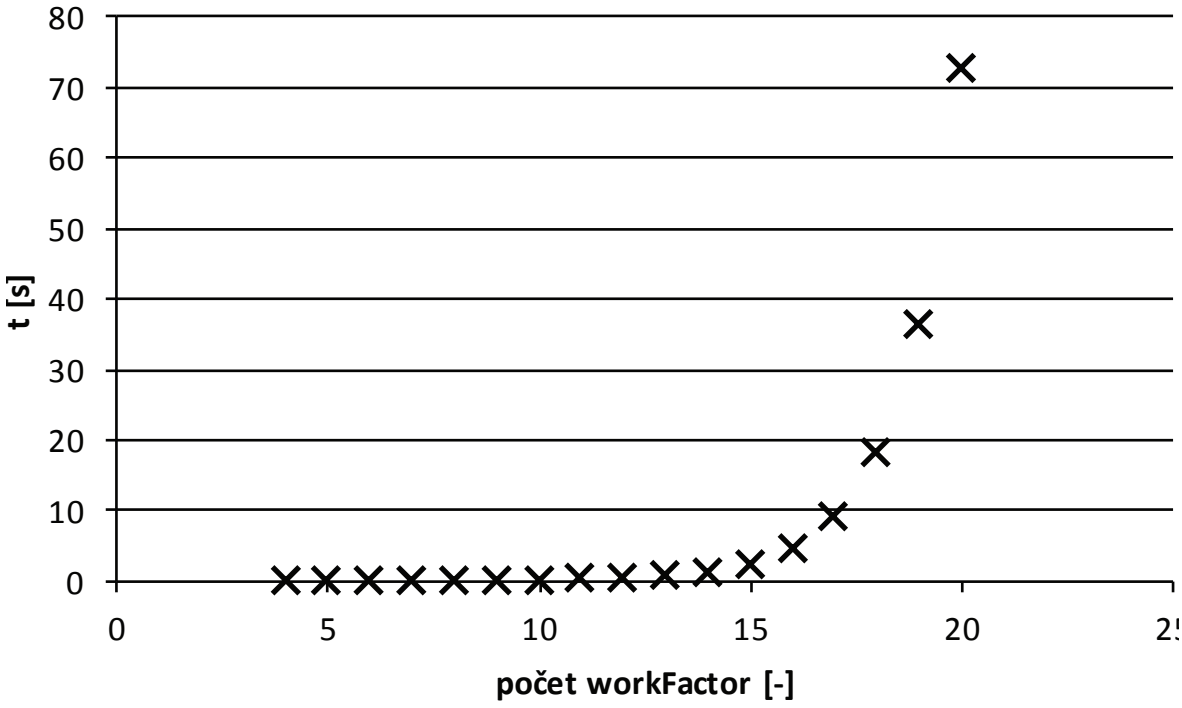
5.2 Zabezpečení proti nepovolenému přístupu

5.2.1 Šifrování hesla při registraci

Nejdříve se zaměříme na samotnou registraci. Při registraci je nutné před uložením hesla do databáze ho důkladně zašifrovat, aby v případě nepovoleného přístupu do databáze nebyly hesla čitelná. Spousta uživatelů používá na internetu jedno až tři hesla a zjištění jediného z nich může mít katastrofální následky. Z tohoto důvodu je zašifrování hesla nezbytné.

Hashovací algoritmy jako je Message-Digest algorithm (MD5) či Secure Hash Algorithm (SHA1) nejsou v dnešní době dostatečně bezpečné. Proto je k důkladnému zašifrování hesla použita funkce **bcrypt** [1]. K hashování vyžaduje nějaký `salt`. V tomto případě byl vygenerován salt pomocí programu KeePass 2 [14] o délce

40 znaků. Další potřebná věc k důkladnému zašifrování je zvolit nějaký hashovací algoritmus – v tomto případě byl zvolen SHA512. A nejdůležitější zbraň je `workFactor`. Ten určuje jak důkladně bude heslo šifrováno. Vliv `workFactor` na výpočetní výkon procesoru je znázorněn na obr. 5.4 převzaté z [16].



Obr. 5.4: Vliv workFactor na výpočetní výkon procesoru

Je vidět, že od určitého stupně začíná čas potřebný k šifrování exponenciálně růst. Je proto důležité vhodně zvolit `workFactor` – v aplikaci je použito `workFactor = 12`. Výstupem je pokaždé řetězec o délce šedesát znaků a pro dvě naprosto stejná hesla se vygeneruje jiný řetězec, takže ani z databáze nelze poznat, že někdo používá heslo stejné jako jiný uživatel.

5.2.2 Autentizace

Celá aplikace je chráněna přihlašovacím systémem, který vyžaduje na ověření zadat email a heslo. Bez správně zadaných údajů není umožněn přístup do administrační části aplikace.

Po vyplnění přihlašovacích údajů a odeslání na server přijde na řadu autentizační skript (obdobný jako při registraci). Při autentizaci je zavolána metoda `authenticate(array $credentials)`, která má jako vstupní parametry zadané přihlašovací údaje. Nejdříve otestuje, zda je zadaná emailová adresa zaregistrována. Pokud není uživatel v databázi nalezen, vrátí systém výjimku a skript se ukončí. Při

správně zadaném emailu se pokračuje na kontrolu, jestli je účet aktivován a poté ověření hesla.

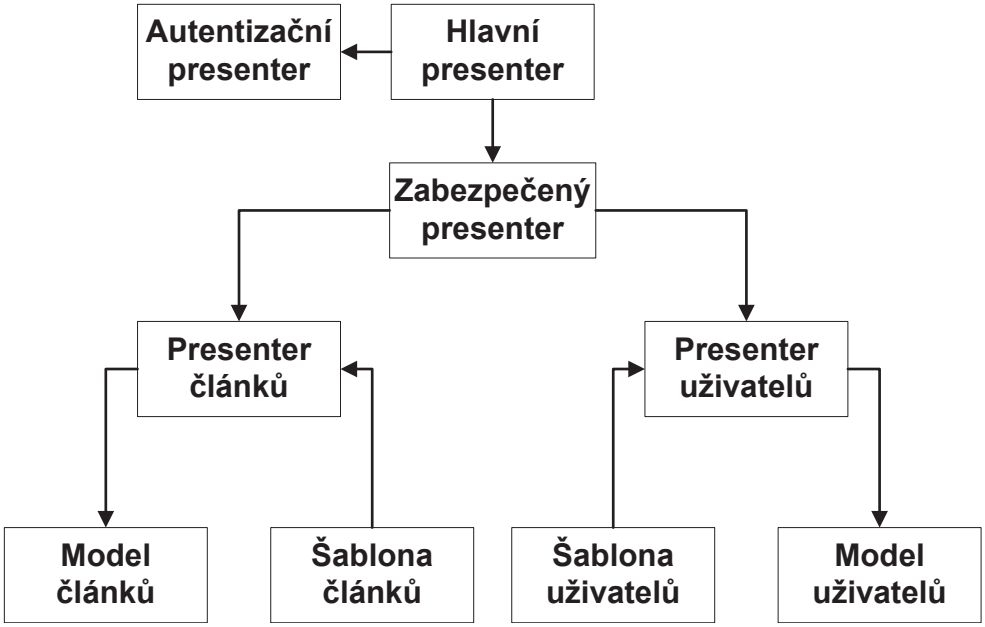
Ověření hesla probíhá pomocí třídy `SolidAuthenticator`, ve které se volá funkce `$this->verifyHash($password, $row->password)`, kde `$password` je heslo zadané uživatelem (nešifrované) a `$row->password` je šifrované heslo z databáze. Tato funkce patří do třídy `Bcrypt`, kde se provede ověření a vrátí hodnotu `TRUE` v případě úspěchu nebo `FALSE` v případě neúspěchu.

Jestliže autentizace úspěšně projde, vrátí metoda zpět identitu uživatele, která je uchována v super globální proměnné `$_SESSION`, ve formě jména uživatele, jeho role a jako třetí parametr může být pole, ve kterém jsou další libovolné údaje.

5.3 Jádro

Na obrázku 5.5 je vidět návrh administrační části systému. Frontend systému je řešen stejně, jen neobsahuje *zabezpečený presenter*.

- *Hlavní presenter (BasePresenter)* rozšiřuje třídu `Nette\Application\UI\Presenter`, která reaguje na události pocházející od uživatele a zajišťuje změny v modelu nebo v pohledu. Hlavní presenter je omezen pouze na nastavení před načtením stránky, jako načtení menu, nastavení výchozího vzhledu a pokud je již uživatel přihlášen, tak načtení uživatelských dat.
- *Zabezpečený presenter (SecuredPresenter)* rozšiřuje třídu *hlavní presenter* o kontrolu přístupových práv do administrační části (ACL). Dále hlídá nečinnost uživatele a v případě dlouhé neaktivity ho z bezpečnostních důvodů odhlásí. Pokud uživatel není přihlášen, přesměruje ho na autorizační presenter.
- *Hlavní šablona (layout)* je výchozí stránka, která je pro všechny další stránky stejná a mění pouze část, která zobrazuje události vyžádané uživatelem. Ty získává přes další šablony, které jsou na to přizpůsobeny.
- *Šablona článků a šablona uživatelů* zobrazují odezvy na události vyvolané uživatelem, které zpracovává presenter ke kterému patří. Ten reaguje na činnost uživatele a podle toho si vyžádá od modelu data z databáze. Samotné presentery vždy rozšiřují *zabezpečený presenter*, výjimku tvoří autentizační presenter. Tím je každý presenter ochráněn proti nepovolenému přístupu nebo proti uživatelům, kteří nemají dostatečná oprávnění pro přístup.



Obr. 5.5: Realizace jádra systému

6 MODUL PRO DOPRAVNÍ PODNIK

Rozšíření z obyčejného redakčního systému na on-line správu „chytré karty“ proběhlo v několika krocích a bylo rozděleno na dílčí části. Veškerá realizace byla dělána podle aktuálních dat Integrovaného dopravního systému Jihomoravského kraje. Ovšem byla brána v úvahu možnost nasazení na jiný integrovaný systém a byla vytvořena určitá univerzálnost. Vytvořit zcela univerzální možnost bylo nemožné, při přechodu na jiný systém jsou tedy nutné úpravy.

6.1 Uživatelská část

K této části mají přístup všichni uživatelé. Získají zde aktuální informace o dění v dopravě, registraci do aplikace, objednání nové karty a následně její správu. Dále je možné na kartu zakupovat nové kupóny, které umožňují využívat služeb dopravního podniku.

6.1.1 Registrace nového uživatele

Registrace do aplikace je jednoduchá a zabere jen pár minut. Pro fyzické osoby stačí vyplnit jen emailovou adresu a poté už jen heslo pro přístup, které je nutné zadat 2x, aby se zamezilo překlepům. V případě registrace společnosti jsou ještě vyžadovány fakturační údaje. Po úspěšné registraci jsou vyplněné údaje zaslány na zadaný email. V případě, že uživatel zapomene heslo pro přihlášení, může si nechat vygenerovat nové, které bude odesláno na registrační email k účtu.

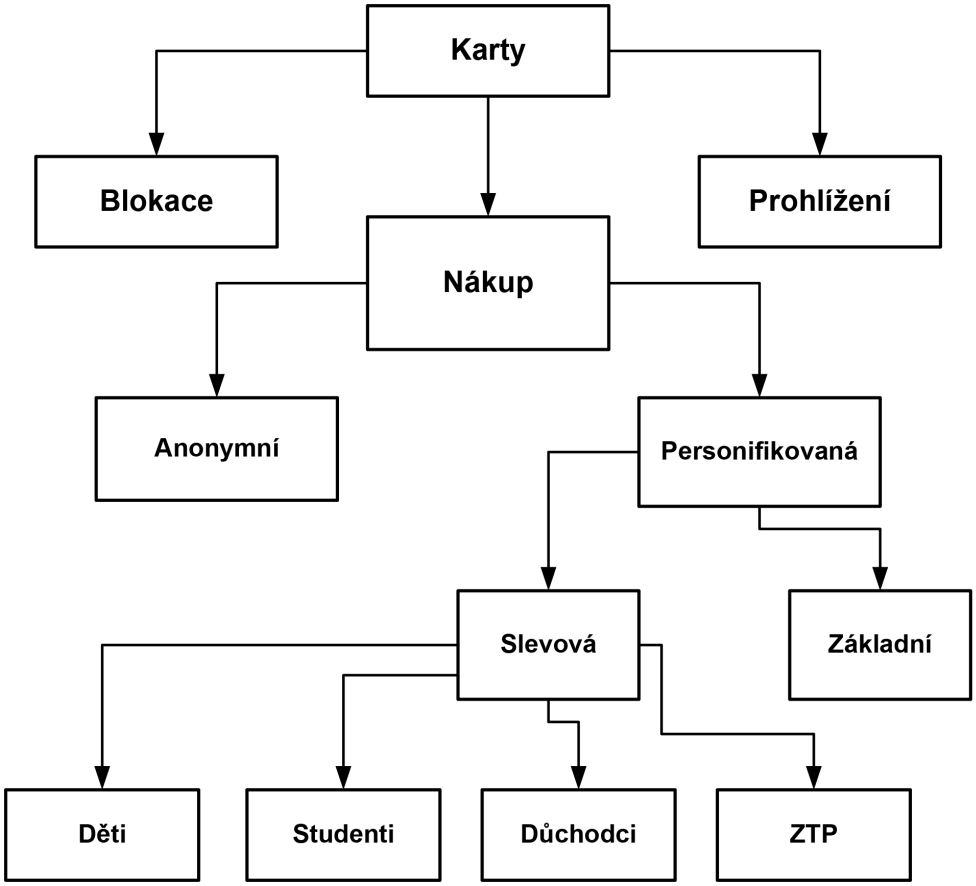
6.1.2 Správa účtu

Registrací do aplikace získá uživatel značné výhody. V nastavení účtu je možné vyplnit další osobní údaje, které se poté mohou použít u objednávek, a nemusí se tak vyplňovat opakovaně všechny údaje.

Další funkce, která se zpřístupní, je přehled o provedených objednávkách a možnost spravovat svoje karty, případně přidávat další. U karet uživatel může v případě ztráty nebo odcizení, kartu zablokovat. V případě uživatelské blokace karty je možné ji aktivovat bez zásahu osoby s vyšším oprávněním.

6.1.3 Karty

Možnost nákupu karet byla realizována s ohledem na možné varianty karet. Rozdělení bylo realizováno podle aktuální nabídky dopravce. Celý proces je znázorněn na schématu na obrázku 6.1.



Obr. 6.1: Schéma návrhu karet

Profil karty a údaje

V první řadě je potřeba vybrat vhodný profil karty. Profily jsou udělány dynamicky a načítají se podle zadání do administrace. Základní rozdělení je na dvě kategorie, kde se rozlišují karty, které jsou dělány přímo pro určitou osobu nebo jako přenosné karty, které jsou anonymní. U anonymní karty je přeskočen další krok objednávky a pokračuje se přímo k volbě doručení karty a platby. Následující krok je zaměřen na získání osobních informací od uživatele, které budou později spojeny s kartou. Jsou požadovány pouze údaje jako je jméno a příjmení a také je potřeba nahrát průkazovou fotografii, která bude s kartou spojena a vytištěna na kartě. V případě výběru profilu, který je omezen věkem, je potřeba zadat i datum narození. S tím je omezen i další krok, kde se volí způsob doručení karty.

Způsob doručení

U volby doručení karty jsou dvě možnosti, a to osobní odběr na některé pobočce nebo možnost nechat si poslat kartu na zvolenou korespondenční adresu. Možnost nechat si poslat kartu na adresu je možné pouze u anonymní karty a základní karty,

kde nejsou uplatňovány žádné slevy. V případě slevových karet je možné vyzvednutí karty pouze na pobočce z důvodu kontroly, že má žadatel nárok na danou slevu (rodným listem, potvrzením o studiu, občanským průkazem atd).

Způsob platby

Jako způsob platby byly zvoleny tři možnosti.

- Hotově,
- bankovním převodem,
- on-line kartou.

Hotově může zákazník platit v případě osobního odběru karty na pobočce. Platba bankovním převodem je velice oblíbená, ale nastává problém, že banky platby pozdržují a platba dojde na účet dopravce dokonce až za dva pracovní dny. V dnešní době je možné u některých bank využít garantované platby, kde platba probíhá přes online bankovnictví, ale všechny údaje jsou předvyplněny a po odeslání je zpětně bankou poskytovateli předána informace o platbě. Jako poslední, a u nás rozšiřující se platba, je pomocí platební karty.

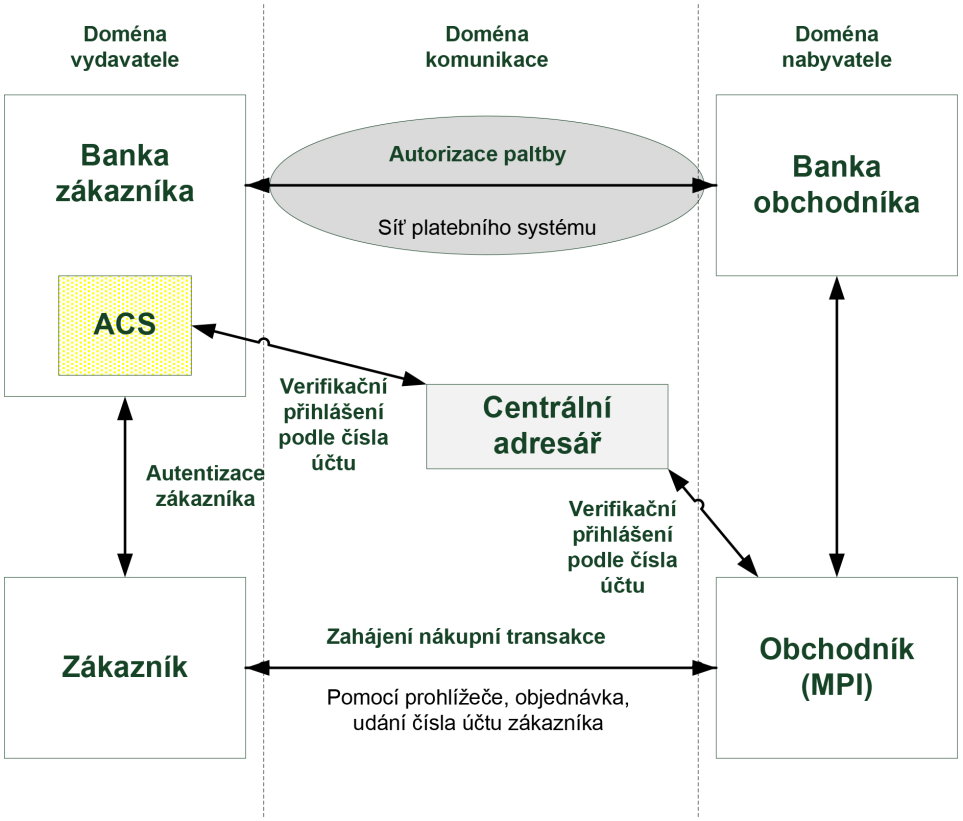
3D Secure

Tato metoda vznikla díky karetním asociacím Visa a MasterCard. Zabezpečení je pomocí SSL/TLS. Tento způsob platby umožňuje kromě ověření samotné karty i ověření identity držitele karty (většinou prostřednictvím SMS kódu zaslaného během platby), což výrazně zvyšuje úroveň zabezpečení transakce a snižuje riziko zneužití karty. Držitel karty zadává údaje o kartě přímo v prostředí banky u které má obchodník zřízen účet, obchodník tedy nemá vůbec přístup ke karetním údajům svých zákazníků. Odtud také název **3-Domain Secure**. Na obrázku 6.2 je znázorněno jak probíhá platba pomocí této metody.

Doména vydavatele - vydavatel kreditní karty je zodpovědný za přihlášení a ověření zákazníka během online transakce, který vlastní jejich kartu.

Doména nabyvatele - zajišťuje aktuální zpracování a ověření, zda je obchodník zapojen do systému 3-D secure.

Doména komunikace - ulehčuje transakci i přes běžně používané protokoly a sdílené systémy.



Obr. 6.2: Struktura 3D Secure [17]

Cílem této platební metody je ochrana obchodníka před neuhrazením závazků klientů. Využívá se http technika „přesměrování“, díky které celá platba probíhá na straně banky obchodníka. Všechny spojení jsou zabezpečeny pomocí SSL/TLS

Do aplikace je tato metoda přidána pomocí dodaného pluginu [13] a připravena k běhu. Do vytvořené továrničky jen stačí přiřadit správné hodnoty od banky včetně soukromého a veřejného klíče a vše bude fungovat jak má.

```
<?php
protected function createComponentWebPay() {
  $wp = new \WebPay;
  $wp->setRequestUrl('https://3dsecure.exemple.com/order.do');
  $wp->setMerchantNumber(123456);
  $wp->setPublicKey(dirname(__FILE__) . '/signing.pem');
  $wp->setPrivateKey(dirname(__FILE__) . '/my.pem', 'heslo');
  $wp->onCreate[] = array($this, 'onCreate');
  $wp->onResponse[] = array($this, 'onResponse');
  return $wp;
}
?>
```

Komponenta reaguje na odpovědi od platební brány a podle toho se chová. V případě úspěšné platby se vrátí kladná odpověď a poté se zavolá metoda `onResponse`:

```
<?php
public function onResponse(\WebPay $webPay,
  \WebPayResponse $response)
{

  $this->context->orderCoupon
          ->webpay($response, $webPay->getParam());
  $this->flashMessage('Platba␣probehla␣v␣poradku,␣dekujeme.');
}
?>
```

Pokud naopak brána vrátí chybovou odpověď (například špatná autorizace, přečerpané limity karty apod.), tak je možné využít metodu `onError`. V aplikaci se metoda `onError` nepoužívá a defaultně se chybová hláška zobrazí jako `flashMessage()` zpráva.

6.1.4 Kupóny

Aby bylo možné využívat kartu, je potřeba zakoupit kupón. V prvním kroku je potřeba zadat číslo karty v případě nepřihlášeného uživatele, a nebo vybrat ze seznamu karet, které jsou přiřazeny k účtu, pod který je uživatel přihlášen. Číslo karty je 18-ti místné unikátní číslo.

Typy kupónů

Typy kupónů jsou závislé na zvoleném typu karty. V našem případě jsou nachystané případy na dlouhodobé předplatní kupóny. Nachystané kupóny jsou na měsíční, čtvrtletní a roční období. Neměl by být problém ani na jednorázové jízdenky, ale u těch nastává otázka, zda se vyplatí.

Volba zón

V této části je nutné navolit požadované zóny, přes které se chceme přepravovat. Je možné vybrat libovolný počet zón, avšak při výběru zóny 100 je výjimka, protože ta nesmí být samotná. Při výběru více jak deseti zón je automaticky vybrána celosíťová jízdenka.

Platnost kupónu a platba

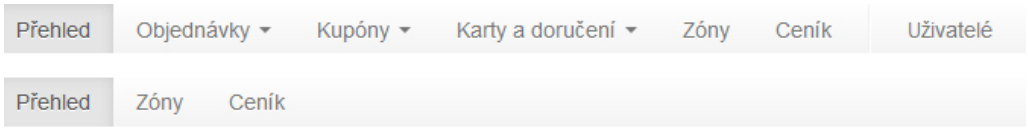
Pomocí mini kalendáře je možné zvolit datum od kdy má kupón platit. Díky tomu je možné si pohodlně zvolit den v týdnu, a to od dne nákupu až měsíc předem. Toto záleží na možnostech dopravce a jeho přepravních podmínkách. Při platbě na bankovní účet je podmínka, že platnost kupónu musí být minimálně tři dny od platby. Je to z důvodu, že bankám převody trvají až tři dny.

Poté následuje souhrn objednávky, kde jsou všechny údaje shrnuty do jedné tabulky a kliknutím na dokončení objednávky se odešle objednávka ke zpracování.

6.2 Administrační část

V administrační části se provádí veškeré nastavení aplikace a přístup je chráněn pomocí přístupových práv. V případě, že uživatel nemá přístup k dané části administrace a i přesto se tam snaží dostat, tak to aplikace rozpozná a vyvolá výjimku s chybovým kódem `HTTP/1.1 403 Forbidden` a zobrazí chybovou stránku k tomu určenou.

Přístupovým právům je přizpůsobeno i administrační menu. Pokud nemá přihlášený uživatel přístup k nějaké sekci administrace, tak se mu v menu vůbec nezobrazí. Na obrázku 6.3 je zobrazen rozdíl mezi jednotlivým zobrazením. Ve vrchní části je menu pro uživatele s plným přístupem a ve spodní části je menu pro vytvořeného uživatele `test@kashpi.cz`, který má přiřazenou roli `test`. Tato role má pomocí ACL nastaven přístup ke správě zón a správě ceníků. Nikam jinam se uživatel v této roli nedostane. Nemá tedy vůbec přístup k částem, které se starají o uživatele, karty, kupóny a celkově o objednávky.

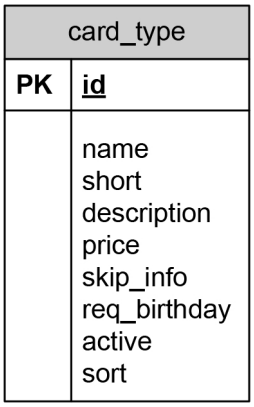


Obr. 6.3: Rozdíl v zobrazení menu podle přístupových práv

6.2.1 Karty a doručení

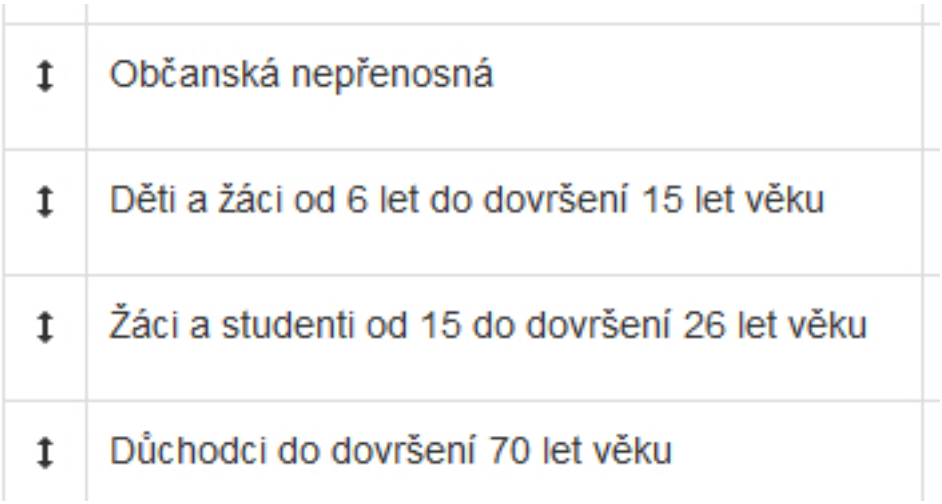
Dynamicky upravovat a přidávat karty je důležitá vlastnost, která vyžadovala důkladné odladění, aby zabrala celý rozsah všech druhů karet. Výsledná tabulka v databázi je znázorněna na obrázku 6.4.

Pole `name` obsahuje název karty v plném znění. Na něj poté navazuje pole `short`, kde je zkrácený název, který je spíše odvozen z názvů jednotlivých ceníků.



Obr. 6.4: Struktura tabulky karet

V `description` je uložen krátký popis. Cena karty je v poli `price`, kde jsou koncové ceny pro zákazníky ukládány s DPH. Položka `skip_info` má jediný význam, a to kvůli anonymním kartám, kde zatrhnutí této hodnoty způsobí, že bude u objednávek karet přeskočen druhý krok, kde se vyplňují osobní údaje na kartu. Další pole `req_birthday` má význam pro karty, kde je vyžadováno datum narození z důvodu slevy (děti, studenti). Položka `active` určuje, zda je karta aktivní pro objednání a zda se zobrazí v možnostech objednat si ji. Například z důvodu zrušení typu této karty apod. Poslední pole je pro přehlednost zobrazených karet. Určuje pořadí, ve kterém se karty zobrazí. Abecední řazení nebylo vhodné, bylo tedy potřeba udělat manuální změnu pořadí. Obyčejné zadávání čísla pořadí by bylo zdlouhavé a nepraktické, proto byla zvolena metoda `drag and drop` znázorněna na obrázku 6.5.



Obr. 6.5: Ukázka řazení karet

Pro tuto možnost byla naprogramována akce, která automaticky přepočítává pořadí položek. Při stisknutí posuvníku se automaticky daný řádek vyjme z tabulky

a je možné jej libovolně vertikálně přesunovat, kde se v tabulce znázorňuje na které pozici se aktuálně nachází. Po uvolnění posuvníku se zařadí položka na pozici, kde se nachází a vyvolá se signál, který odešle aktuální stav řádků v tabulce. Signálová metoda přijme toto pole, které obsahuje `id` karet, seřazené podle řádků v tabulce a pomocí cyklu upraví každý řádek v databázi a přiřadí mu aktuální číslo pozice.

Na karty přímo navazuje možnost jejich doručení. Rozdělují se prakticky na dvě části, a to na vyzvednutí na některé z poboček, a nebo nechat si kartu poslat na určitou adresu. Tyto možnosti jsou podmíněny několika možnostmi. Samotná tabulka pro způsoby doručení je jednoduchá a obsahuje pouze *název, cenu, vyžadování adresy a stav*. Název a cena jsou poměrně jednoznačné. Vyžadování adresy u volby dopravy je z důvodu posílání karty na pobočku. Normálně se při zobrazení poboček nezobrazuje možnost na vyplnění korespondenčních údajů. Při výběru možnosti vyžadovat adresu se při výběru této možnosti u objednávky zobrazí dodatečný formulář na vyplnění další údajů. Stavem je myšleno, zda je tato volba aktivní a zobrazuje se.

Mnohem složitější bylo vymyslet, jak nakombinovat všechny typy karet s možnostmi dopravy. Nejvhodnější bylo vytvořit tabulku dynamickou jak v počtu řádků, tak i v počtu sloupců. O celý tento proces se starala metoda, ve které byly vnořené cykly:

Ukázka kódu pro kombinaci typů karet a jejich možností doručení

```
<?php
public function renderDeliveryCard() {
  $this->template->delivery = $this->context->delivery
    ->findAll()->order('name');

  $cards = $this->context->cardtype->findAll()
    ->order('sort');

  $data = array();
  $i=0;
  foreach($cards as $card) {
    $data[$i]['name'] = $card->name;
    $data[$i]['cid'] = $card->id;
    $j=0;
    foreach($this->context->delivery->findAll()
      ->order('name') as $del)
    {
      if($this->context->cardtypedelivery
        ->findOneBy(array('card_type_id'=>$card->id,
```

```
            'delivery_id'=>$del->id))!== false)
        {
          $is_del = 1;
        }
        else $is_del = 0;
        $data[$i]['pair'][$j] = array('id'=>$del->id,
          'del'=>$is_del);
        $j++;
      }
      $i++;
    }
    $this->template->data = $data;
}
?>
```

Výsledek je ukázán na obrázku 6.6.

	Poštou na adresu	Poštou na adresu doporučeně	Prodejna 1	Prodejna 2	Prodejr 3
Občanská nepřenosná	✓	✓	✓	✓	✗
Děti a žáci od 6 let do dovršení 15 let věku	✗	✗	✗	✗	✗
Žáci a studenti od 15 do	✗	✗	✓	✓	✓

Obr. 6.6: Výsledek kombinace karet a doručení

Vydané karty

Přehled vydaných karet umožňuje vyhledávat karty podle jejich čísla. Jsou zobrazeny údaje o kartě, jako je číslo karty, typ karty, její vlastník nebo alespoň údaje uživatele, který kartu objednal, její platnost a nakonec stav karty. Ten je možné měnit a pomocí jednoduchého odkazu lze kartu aktivovat, deaktivovat, a nebo úplně zablokovat.

6.2.2 Kupóny

Platnosti kupónů byly zvoleny podle aktuální nabídky. Jako jednotka pro platnost byly vybrány dny. V aplikaci jsou aktuálně tři platnosti, a to ve formě kupónu měsíčního, čtvrtletního a ročního. U této metody nastává problém, že všechny platnosti se musí zadávat ve dnech. Měsíční kupón byl nastaven na 30 dní, čtvrtletní na 91 dní a roční na 365 dní. Tímto v určitých měsících nastává problém, protože ne každý měsíc je stejně dlouhý. Ukázka na příkladu bude vhodnější:

```
$test = new \Nette\DateTime('1.1.2013');
dump($test->modify('+91␣days'));

Nette\DateTime(3) {
    date => "2013-04-02␣00:00:00" (19)
    timezone_type => 3
    timezone => "Europe/Prague" (13)
}

$test2 = new \Nette\DateTime('1.1.2013');
dump($test2->modify('+3␣months'));

Nette\DateTime(3) {
    date => "2013-04-01␣00:00:00" (19)
    timezone_type => 3
    timezone => "Europe/Prague" (13)
}
```

V prvním testu uvažujeme hodnotu čtvrt roku jako 91 dní. Při vstupním datu 1. ledna 2013 a úpravy tohoto údaje o 91 dní získáme datum 2. dubna 2013. Při stejném zadání, ale uvažování čtvrt roku jako tři měsíce dostaneme datum 1. dubna 2013. Jde o to, že při hodnotách ve dnech se přičte přesně zadaný počet dní a podle toho nemusí platnost kupónu vycházet dle očekávání. Na druhé straně při využití hodnot v měsících se vždy upraví datum o počet měsíců bez ohledu na to, kolik má daný měsíc dní. Tuhle anomálii by bylo potřeba řešit až při přesném zadání a požadavcích dopravce.

6.2.3 Objednávky

Z důvodu přehlednosti jsou objednávky rozděleny na objednávky karet a objednávky kupónů. U obou případů jsou vizuálně zpracované šablony velmi podobné. Je umožněno vyhledávání podle ID objednávky. U objednávek karet je navíc umožněno vyhledávat podle příjmení uživatele a u objednávek kupónů je navíc vyhledávání podle

čísla karty. Oba druhy mají z důvodu přehlednosti možnost stránkování a možnost si vybrat počet záznamů na stránku.

Objednávky karet

Objednávka karty musí projít přes pověřenou osobu. Je potřeba zkontrolovat správnost vyplněných údajů, jako je kontrola, zda má uživatel nárok na daný typ slevy. V případě studentských slev je nutné kontrolovat tyto údaje ještě před osobním předáním na pobočce. Vidět je i způsob platby, a tím je potřeba zkontrolovat, zda proběhla platba kartou a je už uhrazena nebo v případě platby na účet je potřeba kontrolovat přijetí platby a případně označit objednávku jako zaplacenou.

Objednávka se může nacházet v několika stavech. Po odeslání objednávky uživatelem je označena objednávka jako nová. Po přijetí objednávky dopravcem a zkontrolování údajů je objednávka přesunuta do stavu „zpracovává se“. Všechny tyto stavy, ve kterých se nachází objednávka, vidí uživatel na své kartě objednávek, a tak může mít přehled, v jaké fázi se objednávka nachází. Na uživatelské kartě se vždy zobrazuje stav objednávky ve formátu `stav platby / stav objednávky`. Další stav do kterého může objednávka přejít je „odesláno“ nebo „připraveno na pobočce“. Tyto dva stavy jsou na stejné úrovni a mají stejný význam. Jen je rozdíl v tom, že *odesláno* se nastaví u objednávky, která se posílá poštou a *připraveno na pobočce* na objednávku, která je na osobní odběr. Finální stav objednávky je *dokončeno*, kdy je karta předána nebo doručena zákazníkovi. Celý přehled stavů objednávky je shrnut v následující tabulce:

Tab. 6.1: Stavy objednávek karty

Stav objednávky	**Popis**
nová objednávka	neúplná nebo nezkontrolovaná objednávka
zpracovává se	objednávka byla zkontrolována a předána ke zpracování
odesláno	odeslána zákazníkovi poštou
připraveno na pobočce	připravena na vyzvednutí
dokončeno	předána zákazníkovi

Objednávky kupónů

Objednávky kupónů by měly být z velké části automatické, aby neztratila tato forma kupónů svoji výhodu. Přeci jen jde o pohodlnost a rychlost nákupu. Jako největší výhoda u této formy objednání se dá uvažovat právě absence zásahu další osoby. K tomu je nutná okamžitá platba, kde se jako nejvhodnější uvažuje platba online platební kartou, kde je platba okamžitá a prodejce má zpětnou vazbu od banky, která

potvrdí provedenou platbu a systém může okamžitě kupón aktivovat a zákazník jej může začít používat ihned. Pokud si zákazník zakoupí kupón v dostatečném předstihu a platbu provede pomocí platby převodem na účet, tak je potřeba zásah pověřené osoby. V objednávce je zobrazen typ a číslo karty, ke které se má kupón přiřadit. Dále je typ kupónu a zvolená platnost, od kdy má kupón platit. Z těchto údajů je dopočítán konec platnosti kupónu. Pokud je jako způsob platby zvolen bankovní převod, tak je opět potřeba po přijetí platby, označit objednávku jako zaplacenou. Další důležitá zobrazovaná data jsou zvolené zóny, které si zákazník navolil ke svému kupónu.

Celkově by tato funkcionalita byla lepší, kdyby byla napojená na nějaký firemní systém z důvodu rychlosti a automatičnosti. Také je zbytečné spravovat více stejných dat, protože data o kupónech a kartách musí dopravce vést v nějakém svém programu nebo databázi.

6.2.4 Ceníky

Ceníky [7] jsou hodně individuální a každý dopravce nebo kraj v případě integrovaného dopravního systému má odlišné typy kupónů a rozdělení do jednotlivých kategorií ceníků.

Jako kategorie byly zvoleny dvě:

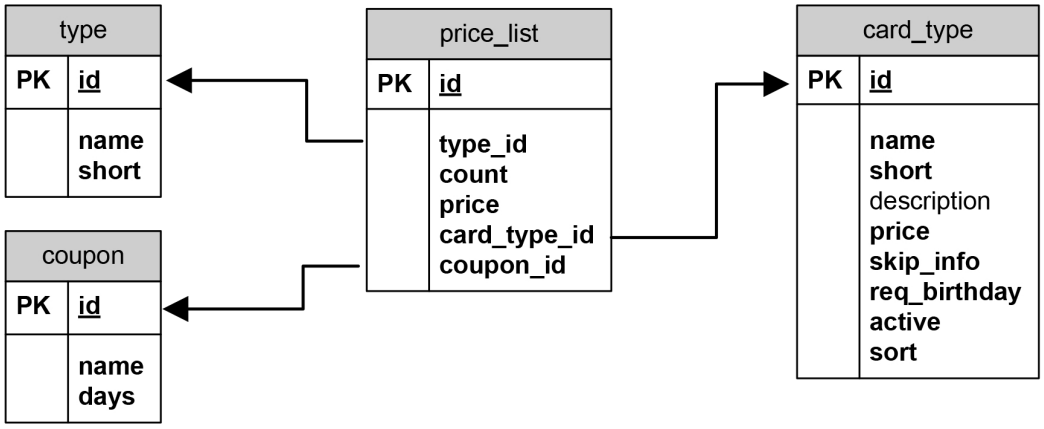
- pro brněnské zóny (100 + 101) s případnou kombinací mimobrněnských zón,
- pro mimobrněnské zóny případně s jednou brněnskou zónou.

Jejich rozdíl je pouze v tom, že pokud kupón obsahuje brněnské zóny 100 a 101, tak jsou určeny jiné ceny, než pro kupóny, které obsahují jiné než brněnské zóny, případně kombinaci pouze s jednou z nich. Oproti jiným krajům nebo městům, kde jsou zónové rozdělení, je pro Jihomoravský kraj vytvořen ceník, který nerozlišuje jednotlivé zóny, ale každá zóna má stejnou „cenu“. Tedy kromě obou brněnských zón. Struktura tabulek ceníků je na následujícím obrázku:

Ceníky byly navrhnuty pouze s ohledem na počet vybraných zón s kontrolou, zda nejsou vybrány obě brněnské zóny. O toto filtrování se stará kousek kódu, který vyhledá v poli shodu na obě zmíněné zóny.

Ukázka kódu pro zjištění výběru obou brněnských zón

```
<?php
foreach($data->zones as $zone) {
  $zones[$zone->zone] = $zone->zone;
}
```



Obr. 6.7: Struktura tabulek ceníků

```
if(in_array(100, $zones) and in_array(101, $zones)) {
  $cenik_type = 1;
}
else {
  $cenik_type = 2;
}
?>
```

Všechny údaje o vybraných zónách jsou v jednom cyklu nakopírovány do pole, kde jako index a hodnota je číslo zóny. Dále je v podmínce využita PHP funkce, která vyhledává hodnoty v poli a vrací `bool` hodnotu podle toho, zda se požadovaná hodnota v poli nachází. V případě nalezení se pomocí logického součinu `AND` rozhodne o splnění podmínky. V případě kladného splnění podmínky se přiřadí ceník s `id = 1`, v opačném případě se přiřadí ceník s `id = 2`. Tímto je provedena první kontrola. Jako další důležité kritérium se kontroluje počet zvolených zón. V případě zvolení více než deseti zón se automaticky zvolí možnost celosíťového kupónu.

Ukázka kódu pro kontrolu počtu zón

```
<?php
$count = count($zones);
if($count > 10 or in_array(0, $zones)) {
  $count = 0;
  $cenik_type = 1;
  $zones = $this->context->zones
    ->findBy(array('zone'=>0))->fetch();
  $data->zones = array(0=>$zones);
  $this->coupon->data['zones'] = array(0=>$zones->id);
  $session['zones'] = array(0=>$zones->id);
```

```
}
else {
  if($count == 1 and (in_array(100, $zones)
      or in_array(101, $zones)))
  {
    $this->flashMessage('V␣pripade␣volby␣zony...','error');
    $this->redirect('step3');
  }
}
?>
```

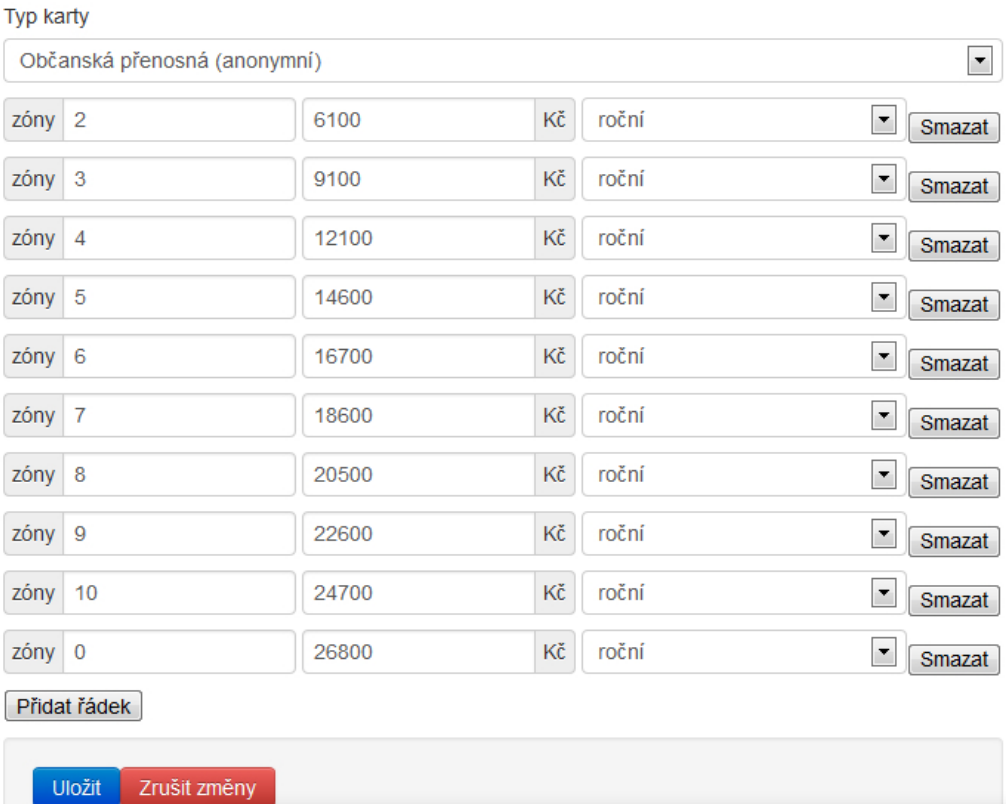
Podmínka kontroluje počet zvolených zón. Je ošetřena i možnost, že uživatel zvolí libovolné zóny a k nim vybere možnost celosíťové varianty, kde celosíťová varianta má zónové označení 0. V případě splnění této podmínky se nastaví typ ceníku na `id = 1` a zjistí se z databáze identifikátor celosíťového kupónu, kterým se přepíší hodnoty hodnoty v polích s uloženými zónami. Tím se bude zobrazovat pouze varianta celosíťového kupónu i v případě vrácení se znovu na výběr zón. Pokud nebude podmínka splněna, tak se kontroluje další podmínka, která má na starost kontrolovat nemožnost vybrat pouze samotnou brněnskou zónu, protože v případě volby pouze brněnské zóny je nutné vybrat i druhou brněnskou zónu.

Z těchto údajů vyplývá struktura tabulky pro ceníky, která je na obrázku 6.7. V tabulce `price_list` jsou uloženy jednotlivé ceníky, kde se zapisuje cena pro stanovený počet zón a vybírají se údaje z okolních tabulek, které určují přesnost výběru. Z tabulky `type` se vybírá, zda je přidávaná hodnota pro typ ceníku brněnského nebo mimobrněnského, z tabulky `card_type` se určuje pro kterou kartu patří a tabulka `coupon` určuje pro jaký časový interval patří. Pomocí těchto všech informací se dá jednoznačně vyfiltrovat požadovaný záznam z tabulky ceníků.

Výsledná stránka s přidáváním a editací ceníků je zobrazena na obrázku 6.8.

Jak je vidět na obrázku, tak možnost editace a přidávání ceníků byla vytvořena značně dynamicky. Je možné libovolně přidávat a odebírat počet řádků v tabulce a tím ovlivňovat výsledný ceník. V případě úprav ceníku je automaticky rozpoznáno, zda byl přidán nový řádek nebo naopak, že byl nějaký odstraněn. Pokud přidáváme nový ceník, tak je vše jednoznačně určeno a každý řádek se přidá. Pokud však přijde na úpravu ceníku, tak je situace složitější, protože se musí rozeznávat rozdíly v přidání dalšího řádku nebo odebrání některého. O celé se stará jedna metoda v modelu, který má na starost práci přímo s databází.

```
<?php
$test = array();
$original = $this->findBy(array('card_type_id'=>$data->card_type_id,
  'type_id'=>$data->type_id));
```



Obr. 6.8: Výpis aktuálního ceníku pro určitý typ karty

```
foreach($original as $row) {
 $test[$row->id] = $row->id;
}
?>
```

V první části se načtou z databáze aktuální řádky a pomocí cyklu se do pole přiřadí všechny identifikátory řádků z daného ceníku. Toto pole slouží k detekci odstraněných řádků (viz dále).

```
<?php
foreach($data->ceniky as $item) {
  if($item->id == "") {
    if($item->count != "") {
            $items['card_type_id'] = $data->card_type_id;
      $items['type_id'] = $data->type_id;
      $items['count'] = $item->count;
      $items['price'] = $item->price;
      $items['coupon_id'] = $item->coupon_id;
      unset($items['id']);
      $values[] = $items;
```

```
        $items = array();
    }
        }
        ...
?>
```

Aktuálně předaná data z formuláře prochází cyklem a kontrolují se nově přidané řádky. V případě nově přidaného řádku, který se pozná podle toho, že nemá vyplněné `id`, které se předává jako skrytá hodnota z formuláře a určuje identifikátor v tabulce v databázi. Pokud se zjistí, že je přidán nový řádek, tak se zkontroluje, zda jsou vyplněné ostatní pole, protože formulář posílá jeden prázdný řádek automaticky, který je nutné odfiltrovat. Pokud projde přes tuto podmínku, tak jsou data zpracována a uložena do jednoho velkého pole, které poté slouží pro multi-insert.

```
<?php
  ...
  else {
    unset($test[$item->id]);
    $items['card_type_id'] = $data->card_type_id;
    $items['type_id'] = $data->type_id;
    $items['count'] = $item->count;
    $items['price'] = $item->price;
    $items['coupon_id'] = $item->coupon_id;
    $items['id'] = $item->id;
    $this->update($items);
  }
}
...
?>
```

Tato část podmínky nastane v případě, že se nejedná a nový řádek, ale o úpravu stávajícího. V této části se upravují data v databázi řádek po řádku. Při splnění podmínky se hned na začátku odstraní identifikátor řádku z původní databáze před úpravami, a tím se způsobí, že se postupně odstraní všechna `id`, která byla upravena. Všechna `id`, která v poli zůstanou, znamenají, že byl daný řádek z formuláře odstraněn.

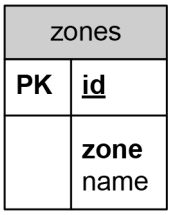
```
<?php
...
if(count($values) > 0) $this->insert($values);
  if($test) {
    foreach($test as $k=>$v) {
      $this->delete($k);
```

```
    }
  }
  return;
}
?>
```

V poslední části je zkontrolováno, zda byl vložen nějaký nový řádek. Pokud ano, tak podmínka, která kontroluje počet hodnot v poli `$values`, je splněna a vykoná se příkaz na vložení dat do databáze. Všechna data jsou vložena najednou z jednoho velkého pole, a to z důvodu ušetření výpočetního výkonu. Nakonec se provede ověření, zda byl odstraněn nějaký řádek z ceníku. Pomocí kontroly jestli jsou v poli `$test` nějaká data. V případě, že ano, tak se v cyklu provede mazání řádek po řádku pomocí identifikátoru řádku z tabulky.

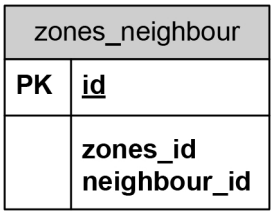
6.2.5 Zóny

Na zóny byla vytvořena tabulka, která je znázorněna na obrázku 6.9. Do sloupce `zone` se vkládají přímo čísla zón a do sloupce `name` jejich slovní označení pro lepší rozpoznání (např. podle města). Čísla zón odpovídají označení, jediná výjimka nastává pro označení celosíťové zóny, které nese označení *0*.



Obr. 6.9: Struktura tabulky zón

Další problém, který nastává u zón je, že je potřeba kontrolovat nebo nějak zabezpečit možnost vybrat zóny, které spolu nesousedí. K tomuto účelu byla vytvořena tabulka v databázi.



Obr. 6.10: Struktura tabulky na kontrolu sousedních zón

Do této tabulky se ukládají jednotlivé zóny a k nim vždy do páru identifikátory sousedních zón. Tím je umožněno vybrat pouze zóny, které spolu sousedí. Zabrání

se tím zneužití, jako třeba vynechání některé zóny úmyslně, a nebo nepozornosti při výběru zón.

Jednoduchou tabulku vyvažuje složitější kód.

```
<?php
$zones = $this->context->zonesneighbour
  ->findBy(array('zones_id'=>$id));
$zones2 = $this->context->zonesneighbour
  ->findBy(array('neighbour_id'=>$id));
$ids=array();
foreach($zones as $row) {
  $ids[] = $row->neighbour_id;
}
foreach($zones2 as $row) {
  $ids[] = $row->zones_id;
}
if($ids) {
  $form = $this->getComponent('neighbour');
  $form->setDefaults(array('zones'=>$ids));
}
?>
```

V první řadě se postupně z tabulky načtou všechna data související s vybranou zónou. Nejdříve se vyberou data podle zón a poté podle sousedů. Oba takto získané objekty jsou pomocí cyklu uloženy postupně do stejného pole, čímž se odstraní duplicity. Dále se toto pole použije jako výchozí hodnoty do formuláře, který se tímto předvyplní. Touto metodou je zajištěno, že když u jedné zóny vyberete souseda, tak se při úpravách sousední zóny zjistí, že již je jeho `id` někde použito a využije se. Tím je částečně redukován počet záznamů v databázi.

Výběr sousedních zón je graficky udělán velmi příjemně a je z něj okamžitě jasné, které zóny jsou vybrané a které naopak nejsou.

Při odeslání formuláře jsou data z této zóny nejdříve odstraněna, a poté nahrána aktuální. Vše je patrné z následujícího kódu:

```
<?php
$values = array();
$item = array();
foreach($data->zones as $k=>$v) {
  $item['zones_id'] = $data->id;
  $item['neighbour_id'] = $v;
  $values[] = $item;
  $item = array();
```

Obr. 6.11: Ukázka výběru sousedních zón

```
}
$this->findBy(array('zones_id'=>$data->id))->delete();
$this->insert($values);
?>
```

Odeslaná data jsou postupně v cyklu přiřazena do jednoho velkého pole. Poté se z databáze odstraní původní hodnoty a pomocí jednoho multi-insertu se nahrají data nová.

Na straně objednávek jsou data kontrolovaná ihned po odeslání formuláře se zónami.

```
<?php
foreach($data->zones as $k=>$v) {
  $ids2[] = $v;
}
$zone1 = $this->context->zonesneighbour
  ->findBy(array('zones_id'=>$ids2));
$zone2 = $this->context->zonesneighbour
  ->findBy(array('neighbour_id'=>$ids2));

$ids=array();
foreach($zone1 as $row) {
  $ids[$row->neighbour_id] = $row->neighbour_id;
}
foreach($zone2 as $row) {
 $ids[$row->zones_id] = $row->zones_id;
}
foreach($data->zones as $k=>$v ) {
  if(!in_array($v, $ids)) {
```

```
            $form->addError('Kombinace techto zon neni mozna');
            return;
        }
    }
?>
```

Z přijatých zón je vytvořeno jedno pole, které slouží pro filtrování dat z tabulky, kde jsou uloženy data o sousedících zónách. Nejprve se filtruje podle identifikátoru zón, a poté podle identifikátoru sousedů. Je vytvořeno prázdné pole, do kterého se průchodem cyklu uloží identifikátory nejdříve z prvního dotazu a poté i identifikátory z druhého dotazu s tím, že jsou duplicitní identifikátory odstraněny. Poté jsou v cyklu procházeny zóny z formuláře a jejich identifikátory kontrolovány na shodu v poli všech sousedních zón.

7 ZÁVĚR

V této práci jsou srovnány nejznámější PHP frameworky. Všechny zde uvedené frameworky podporují moderní technologie včetně podpory více databází, MVC modelu, podpory PHP 5 a dnes využívané technologie AJAX. Jako vhodní kandidáti byly PHP frameworky Nette a Zend. Nová verze Nette 2 má mnoho komponent, které usnadňují práci. Příchod Zend frameworku 2 byl dlouho očekávaný a velmi se zlepšily paměťové nároky a celkově působí lepším dojmem. Také zavedení jmenných prostorů vedlo k zpřehlednění. Oba frameworky byly otestovány a z testu vycházel lépe Nette framework, i když výsledky byly vyrovnané.

Při návrhu byl kladen důraz na jednoduchost a bezpečnost. Snaha byla o co nejvíce optimalizovanou databázi a bezpečnou autentizaci. Při realizaci modulu pro dopravní podnik byl kladen důraz na co největší univerzálnost a možnost využití pro jakéhokoli dopravce.

V uživatelské části byl realizován objednací proces pro nákup chytré karty a následně možnost zakoupit na kartu kupóny umožňující přepravu. Typy karet a platnosti kupónů byly zvoleny podle aktuálních podmínek Integrovaného dopravního systému Jihomoravského kraje. Přičemž funkce starající se o tyto možnosti byly navrhnuty univerzálně, aby je bylo možné použít bez velkých změn na jiný dopravní podnik. Po přihlášení má každý uživatel přehled jak o svých objednávkách, tak i o vydaných kartách a platnostech kupónů. Nezapomnělo se ani na rychlou a bezpečnou platbu, která je u tohoto typu služby samozřejmostí. Využito bylo i moderní platební metody pomocí 3D secure. Celý tento systém by bylo vhodné napojit přímo na databázi dopravce, aby se nemusela získaná data zpracovávat dvakrát.

V administrační části byl kladen důraz na jednoduchost a přehlednost při zadávání nových nebo upravování stávajících ceníků. V části objednávek je umožněno vyhledávat objednávky podle různých kritérií. Ve správě zón je umožněno přidávání nových zón, upravování stávajících a ošetření výběru pouze sousedící zón. Tím se zamezí výběru zón, které spolu nesousedí nebo zapomene-li se zóna vybrat.

LITERATURA

[1] Bcrypt. [online].
URL http://bcrypt.sourceforge.net/

[2] DANĚK, P.: Velký test PHP frameworků. [online], c2008 [cit. 7. 12. 2012].
URL http://www.root.cz/clanky/velky-test-php-frameworku-2008/

[3] Free Software Foundation: GNU General Public License. [online], c2007 [cit. 10. 12. 2012].
URL http://www.gnu.org/licenses/gpl.html

[4] Grudl, D.: Texy! [online].
URL http://texy.info/

[5] GRUDL, D.: Nette framework. [online], c2008 [cit. 7. 12. 2012].
URL http://nette.org

[6] HUNTER, M.: Tidying the House: The MVPC Software Design Pattern. [online], c2006 [cit. 9. 12. 2012].
URL http://www.martinhunter.co.nz/articles/MVPC.pdf

[7] IDS JMK: Integrovaný dopravní systém Jihomoravského kraje. [online], [cit. 1. 5. 2013].
URL http://www.idsjmk.cz/

[8] KAŠPAREC, M.: *Realizace modulárního webového systému: bakalářská práce.* Brno:Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2011.

[9] MySQL: MySQL manual 5.5. [online], c2011 [cit. 10. 12. 2012].
URL http://dev.mysql.com/doc/refman/5.5/en/index.html

[10] Open Source CMS. [online].
URL http://www.opensourcecms.com/scripts/show.php?catid=1&category=CMS%20/%20Portals

[11] PHP: PHP manual. [online], c2011 [cit. 9. 12. 2012].
URL http://cz.php.net/manual/en/

[12] PHPFrameworks.com: PHP Frameworks. [online], c2007 [cit. 18. 11. 2012].
URL http://www.phpframeworks.com/

[13] PROCHÁZKA, P.: Webpay. [online], [cit. 1. 5. 2013].
URL http://petrp.cz/nette-addons/webpay

[14] Reichl, D.: KeePass 2. [online].
URL http://keepass.info/

[15] Super svět: Co je to redakční systém? [online], [cit. 1. 5. 2013].
URL http://www.supersvet.cz/view.php?cisloclanku=2005050501

[16] Wildly Inaccurate: Bcrypt: Choosing a Work Factor. [online], [cit. 10. 12. 2012].
URL http://wildlyinaccurate.com/bcrypt-choosing-a-work-factor

[17] ZEMAN, V.: Elektronické platební systémy, přednáška z předmětu Kryptografie.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACL	Access Control List
AJAX	Asynchronous JavaScript and XML
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CSRF	Cross-site request forgery
DRY	Don't repeat yourself
GNU-GPL	GNU General Public License
IP	Internet Protocol
KISS	Keep it short and simple
MD5	Message-Digest algorithm
MVC	Model-View-Controller
MVP	Model-View-Presenter
ODBC	Open Database Connectivity
ORM	Object-relational mapping
PDF	Portable Document Format
PDO	PHP Data Objects
PHP	PHP: Hypertex Preprocessor
RS	Redakční systém
RSS	Really Simple Syndication
SHA1	Secure Hash Algorithm
SQL	Structured Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Unique Resource Locator
XSS	Cross-site scripting

SEZNAM PŘÍLOH

A UKÁZKY WEBOVÉHO SYSTÉMU

A.1 Online verze

adresa URL: http://dp.kashpi.cz
URL do administrace: http://dp.kashpi.cz/admin

Uživatel v roli admin:

- email: admin@kashpi.cz
- heslo: admin

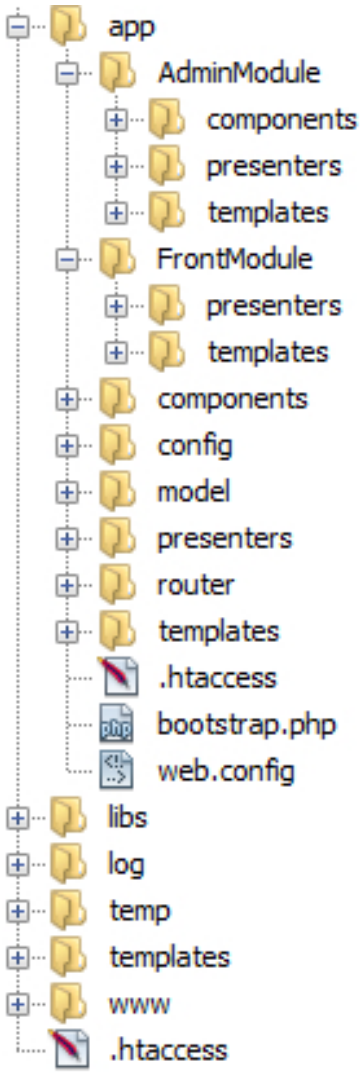
Uživatel s omezeným oprávněním (pouze ceníky a zóny):

- email: test@kashpi.cz
- heslo: admin

Uživatel bez role (role „registrovaný“)

- email: user@kashpi.cz
- heslo: user123

A.2 Adresářová struktura



Obr. A.1: Adresářová struktura

B OBSAH PŘILOŽENÉHO CD

Na přiloženém CD je adresář `www` se zdrojovými kódy webového systému a adresář `mysql` se skriptem pro vytvoření databáze. V kořenovém adresáři je soubor s textem diplomové práce.

V adresáři `www/app/config` je nutné upravit soubor `config.neon` tak, aby se webová aplikace mohla připojit k databázovému serveru MySQL.

Celý systém byl vyvíjen a testován na webovém serveru Apache 2.4.4 (Win64) s rozšířením PHP 5.4.5. Jako databázový server byl použit MySQL 5.5.31.